

Shortest Paths I: Properties, Dijkstra's Algorithm, Breadth- first Search

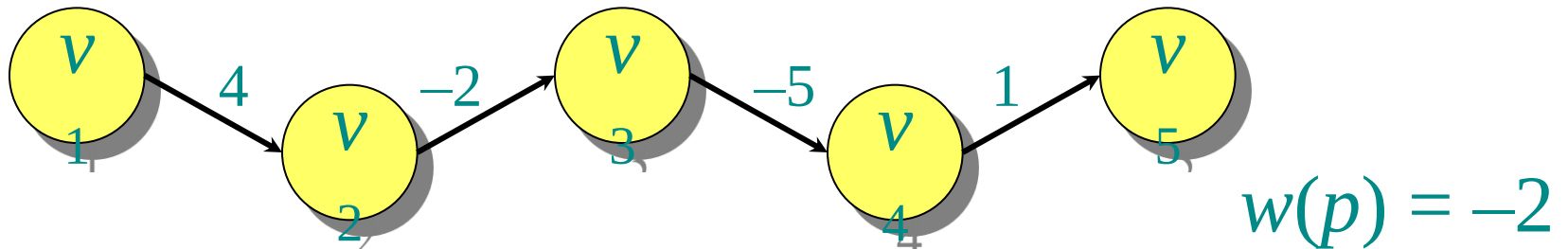
Lecture 14

Paths in graphs

Consider a digraph $G = (V, E)$ with edge-weight function $w : E \rightarrow \mathbb{R}$. The **weight** of path $p = v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_k$ is defined to be

$$w(p) = \sum_{i=1}^{k-1} w(v_i, v_{i+1}).$$

Example:



Shortest paths

A *shortest path* from u to v is a path of minimum weight from u to v . The *shortest-path weight* from u to v is defined as

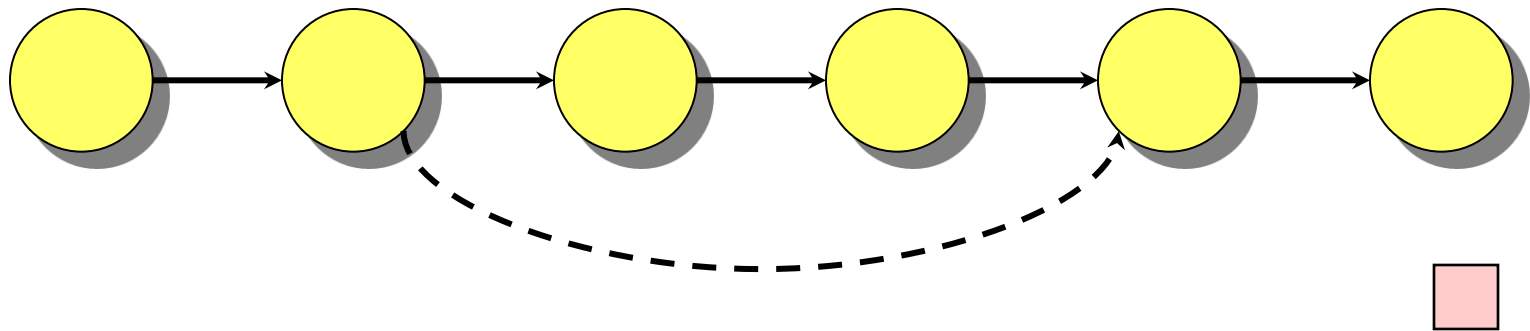
$$\delta(u, v) = \min\{w(p) : p \text{ is a path from } u \text{ to } v\}.$$

Note: $\delta(u, v) = \infty$ if no path from u to v exists.

Optimal substructure

Theorem. A subpath of a shortest path is a shortest path.

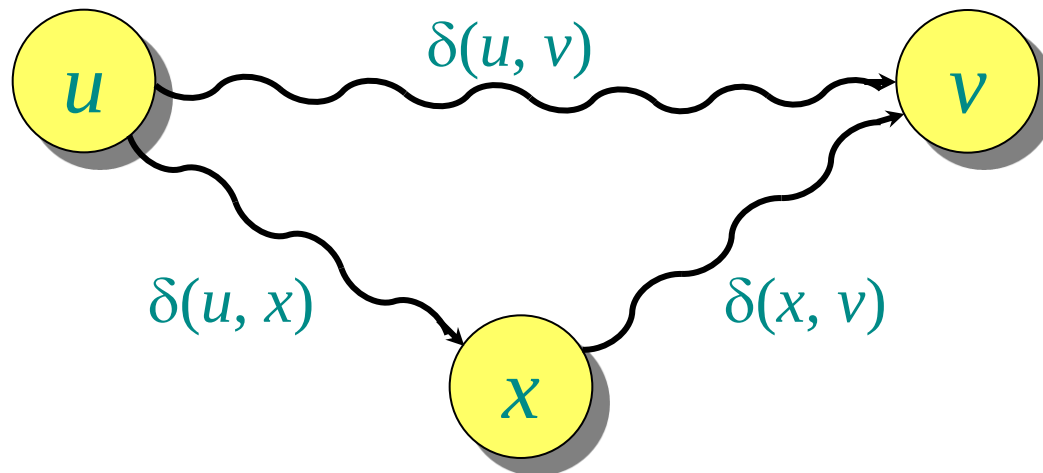
Proof. Cut and paste:



Triangle inequality

Theorem. For all $u, v, x \in V$, we have
$$\delta(u, v) \leq \delta(u, x) + \delta(x, v).$$

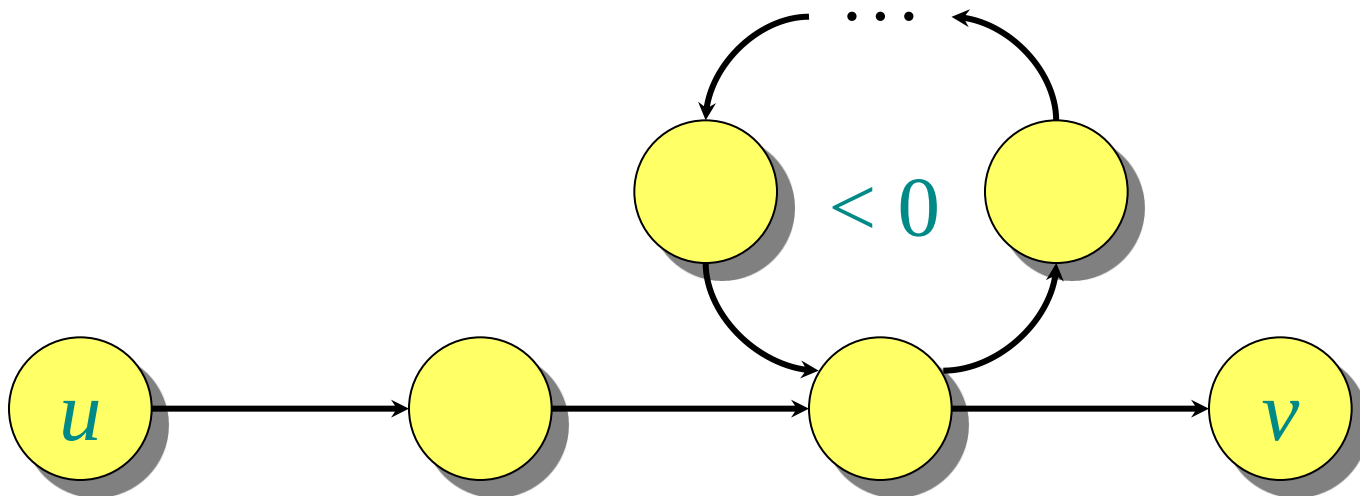
Proof.



Well-definedness of shortest paths

If a graph G contains a negative-weight cycle, then some shortest paths may not exist.

Example:



Single-source shortest paths

Problem. From a given source vertex $s \in V$, find the shortest-path weights $\delta(s, v)$ for all $v \in V$.

If all edge weights $w(u, v)$ are *nonnegative*, all shortest-path weights must exist.

IDEA: Greedy.

1. Maintain a set S of vertices whose shortest-path distances from s are known.
2. At each step add to S the vertex $v \in V - S$ whose distance estimate from s is minimal.
3. Update the distance estimates of vertices adjacent to v .

Dijkstra's algorithm

$d[s] \leftarrow 0$

for each $v \in V - \{s\}$

do $d[v] \leftarrow \infty$

$S \leftarrow \emptyset$

$Q \leftarrow V$

▷ Q is a priority queue maintaining $V - S$

while $Q \neq \emptyset$

do $u \leftarrow \text{EXTRACT-MIN}(Q)$

$S \leftarrow S \cup \{u\}$

for each $v \in \text{Adj}[u]$

do if $d[v] > d[u] + w(u, v)$

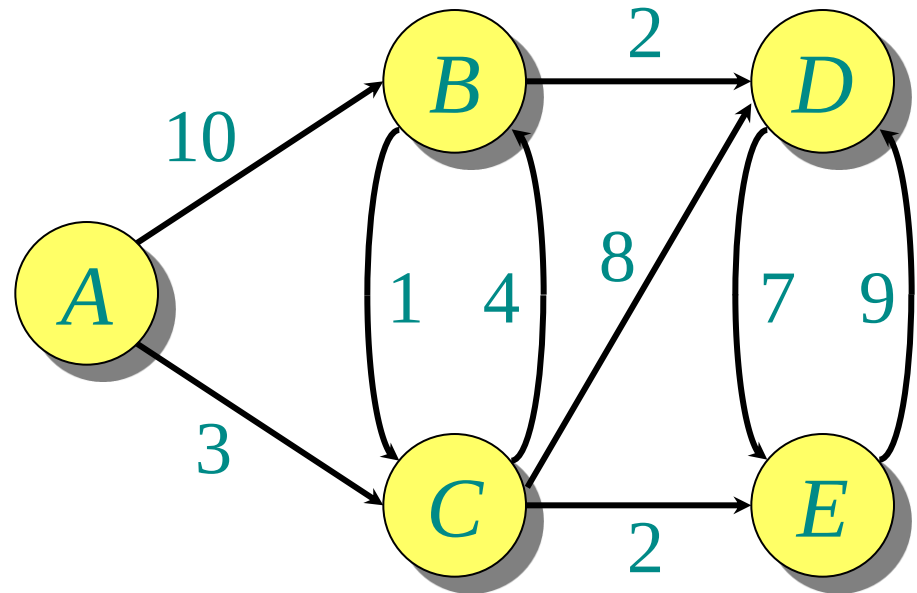
then $d[v] \leftarrow d[u] + w(u, v)$

*relaxation
step*

Implicit DECREASE-KEY

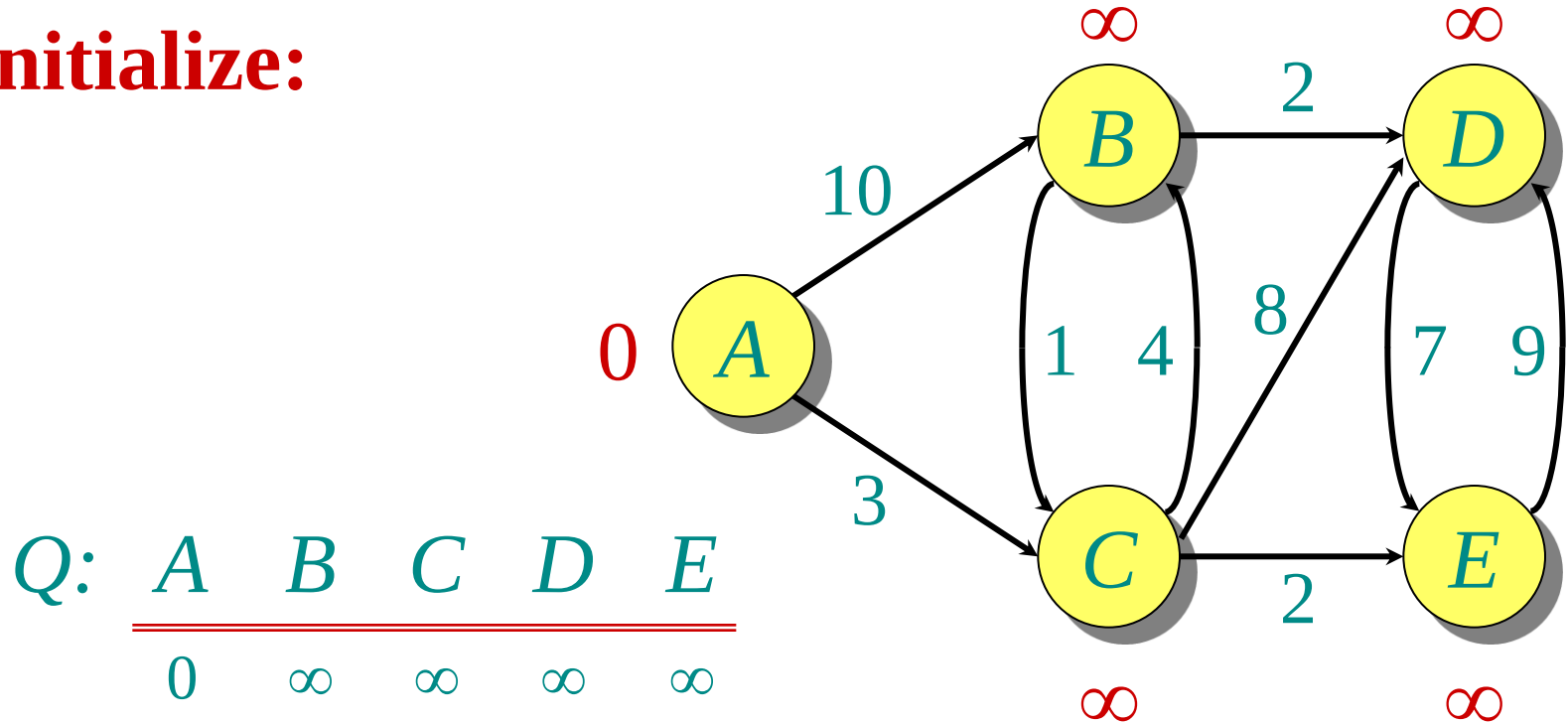
Example of Dijkstra's algorithm

Graph with
nonnegative
edge weights:



Example of Dijkstra's algorithm

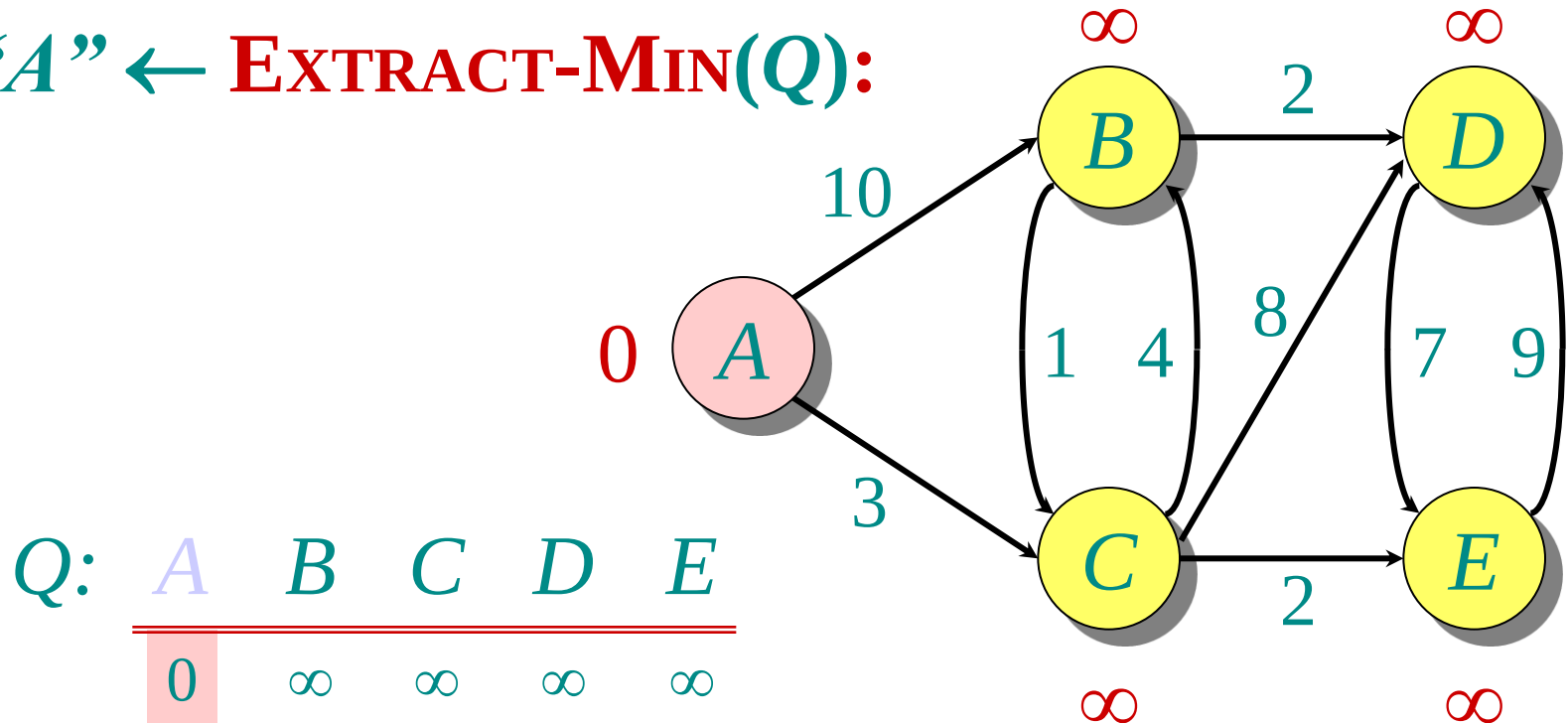
Initialize:



S: {}

Example of Dijkstra's algorithm

“A” $\leftarrow$ **EXTRACT-MIN**(Q):



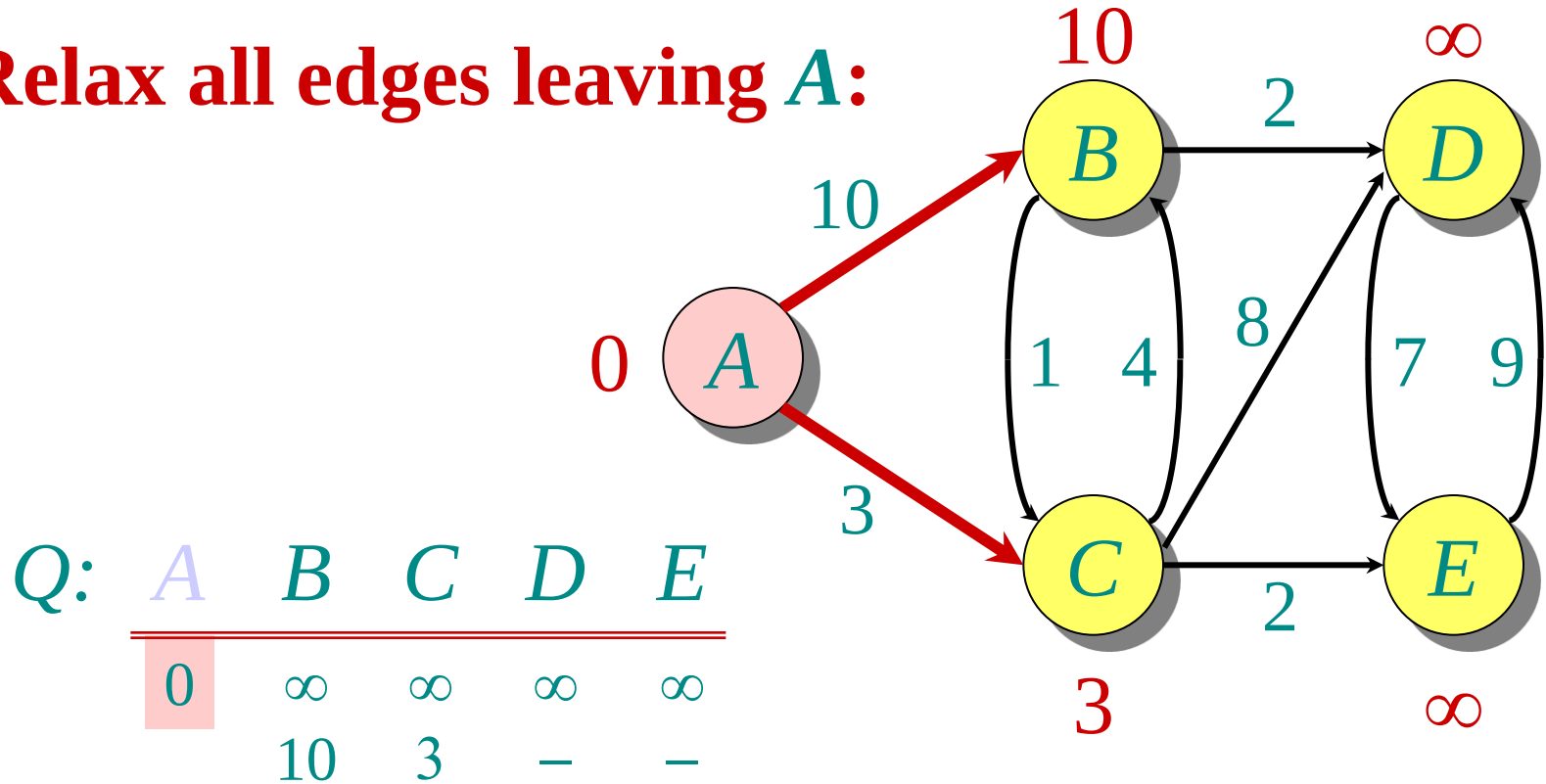
Q:

A	B	C	D	E
0	∞	∞	∞	∞

S: { A }

Example of Dijkstra's algorithm

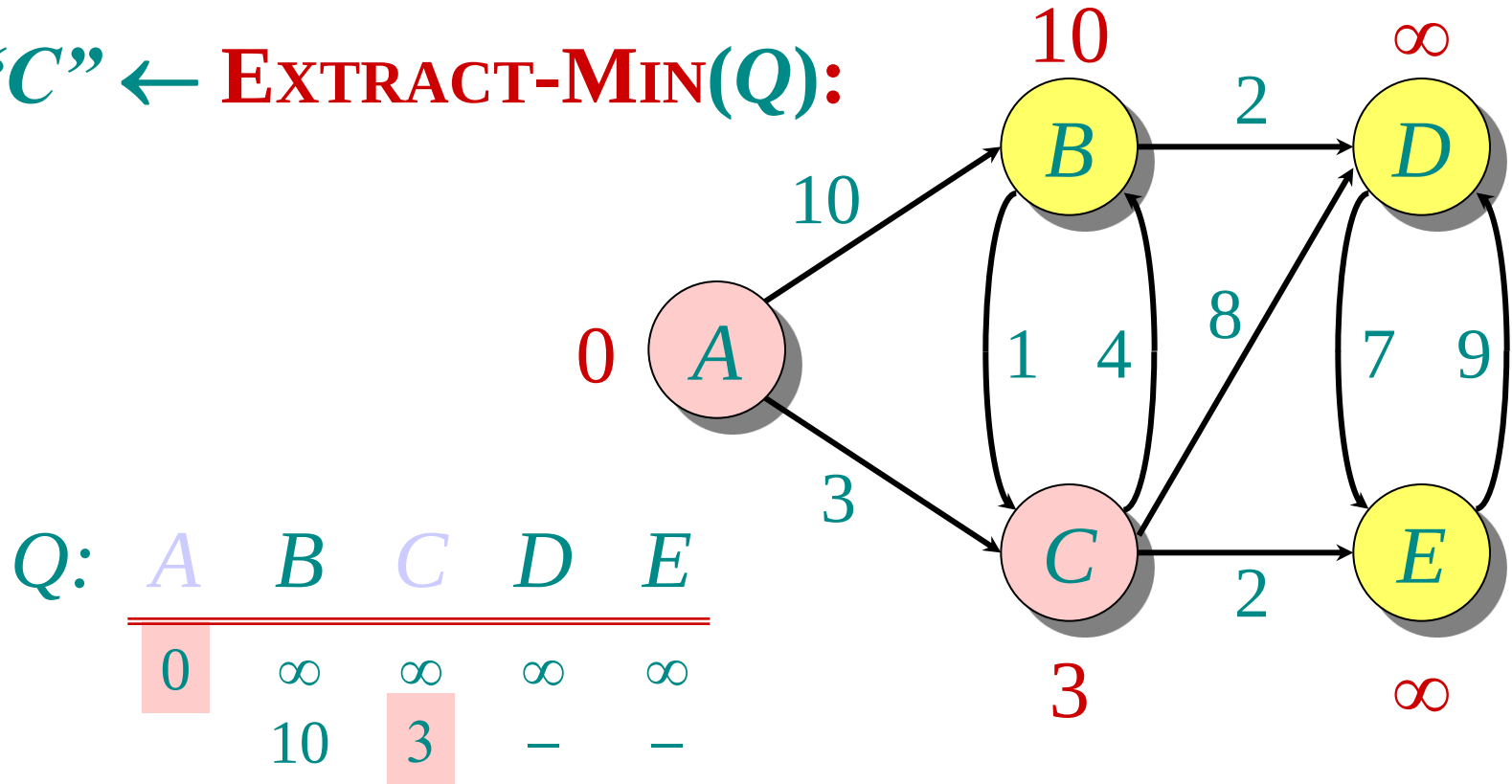
Relax all edges leaving **A**:



S: { A }

Example of Dijkstra's algorithm

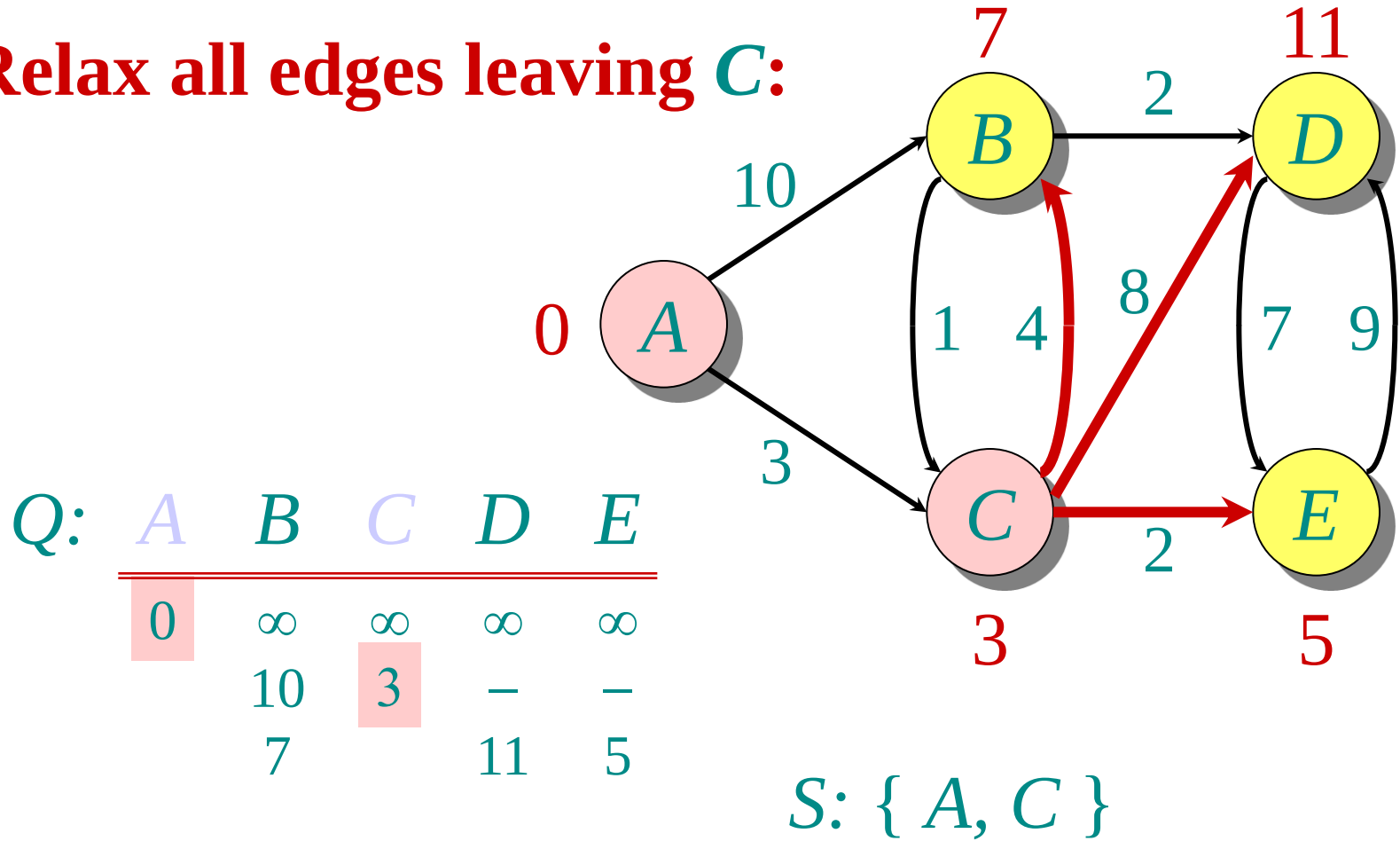
“C” $\leftarrow$ **EXTRACT-MIN**(Q):



S: { A, C }

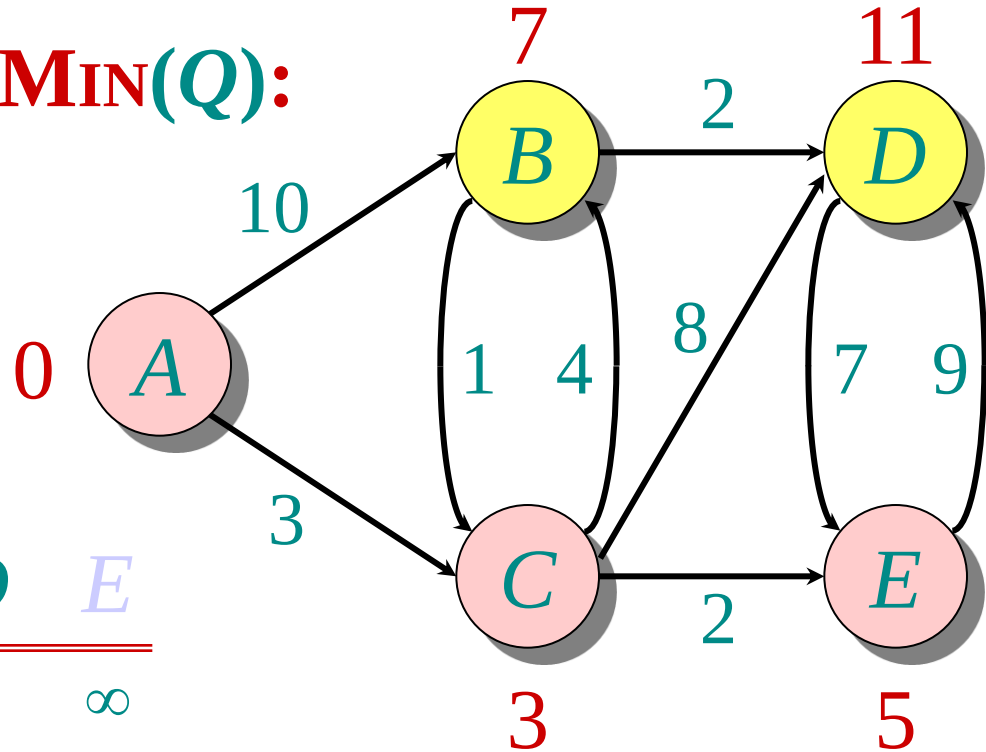
Example of Dijkstra's algorithm

Relax all edges leaving **C**:



Example of Dijkstra's algorithm

E ← **EXTRACT-MIN(Q):**

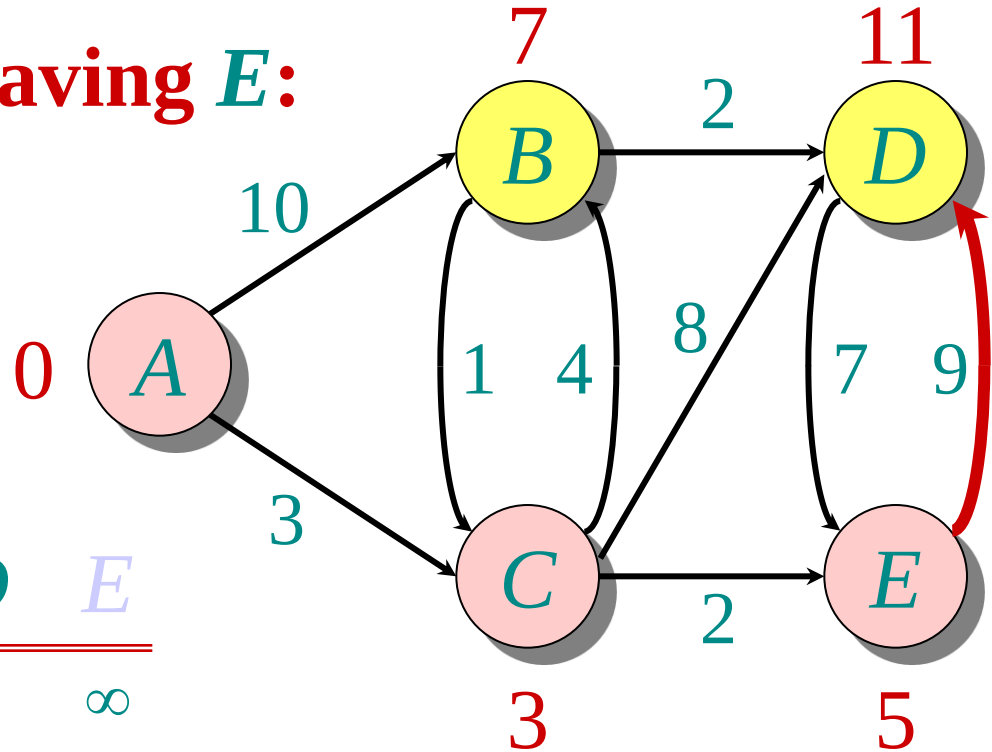


Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	—	—
		7		11	5

S: { *A*, *C*, *E* }

Example of Dijkstra's algorithm

Relax all edges leaving E :



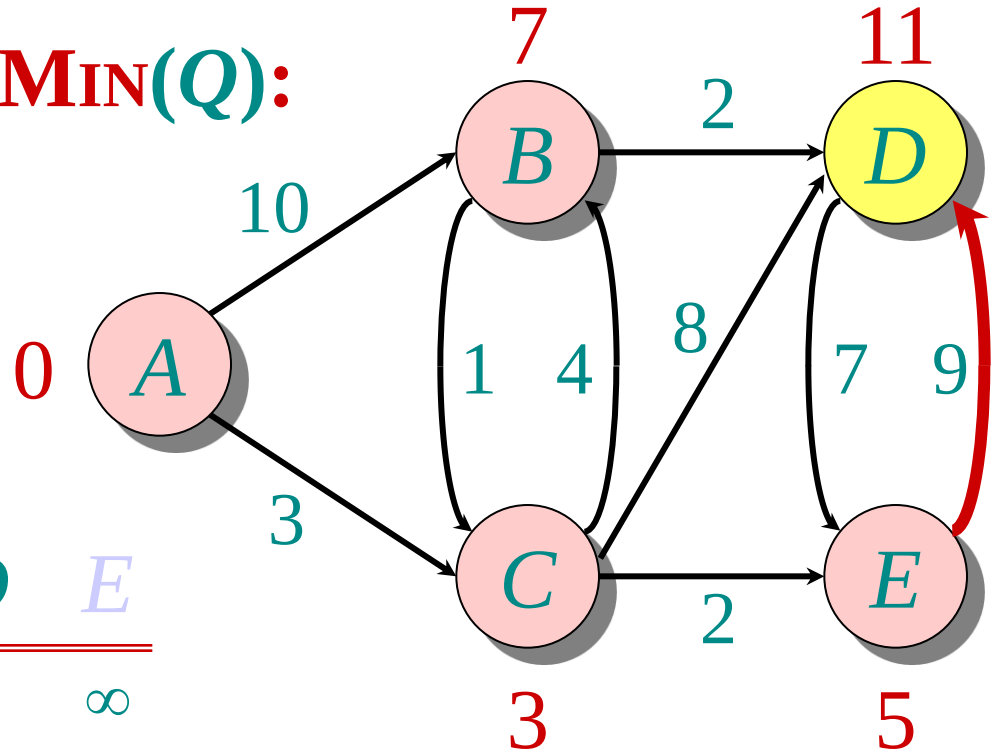
Q:

A	B	C	D	E
0	∞	∞	∞	∞
	10	3	∞	∞
	7		11	5
	7		11	

$S: \{ A, C, E \}$

Example of Dijkstra's algorithm

"B" ← EXTRACT-MIN(Q):



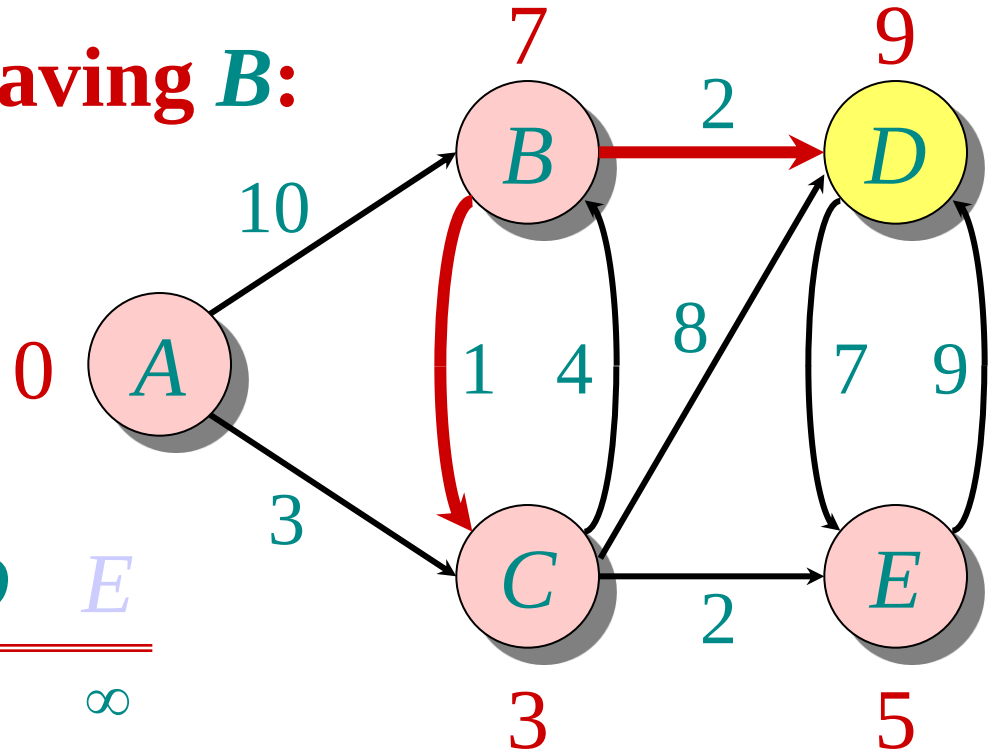
Q:

A	B	C	D	E
0	∞	∞	∞	∞
	10	3	∞	∞
	7		11	5
	7		11	

$S: \{ A, C, E, B \}$

Example of Dijkstra's algorithm

Relax all edges leaving **B**:



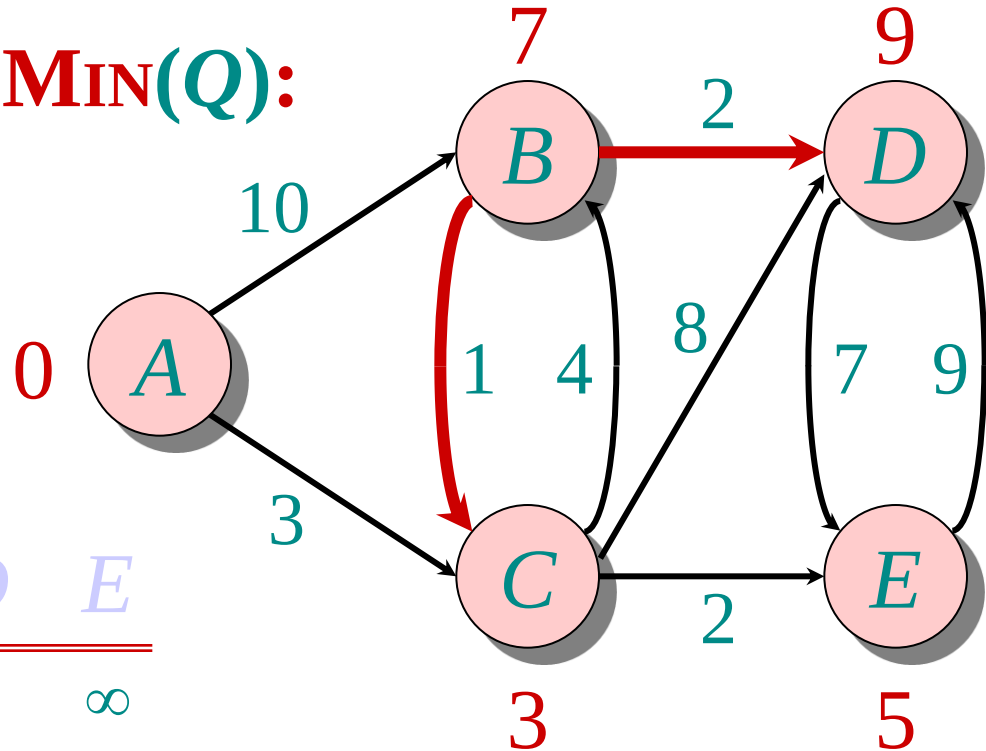
Q:

A	B	C	D	E
0	∞	∞	∞	∞
	10	3	∞	∞
	7		11	5
	7		11	
			9	

$S: \{ A, C, E, B \}$

Example of Dijkstra's algorithm

"D" $\leftarrow$ **EXTRACT-MIN(Q):**



Q:

A	B	C	D	E
0	∞	∞	∞	∞
	10	3	∞	∞
	7		11	5
	7		11	
			9	

$S: \{ A, C, E, B, D \}$

Correctness — Part I

Lemma. Initializing $d[s] \leftarrow 0$ and $d[v] \leftarrow \infty$ for all $v \in V - \{s\}$ establishes $d[v] \geq \delta(s, v)$ for all $v \in V$, and this invariant is maintained over any sequence of relaxation steps.

Proof. Suppose not. Let v be the first vertex for which $d[v] < \delta(s, v)$, and let u be the vertex that caused $d[v]$ to change: $d[v] = d[u] + w(u, v)$. Then,

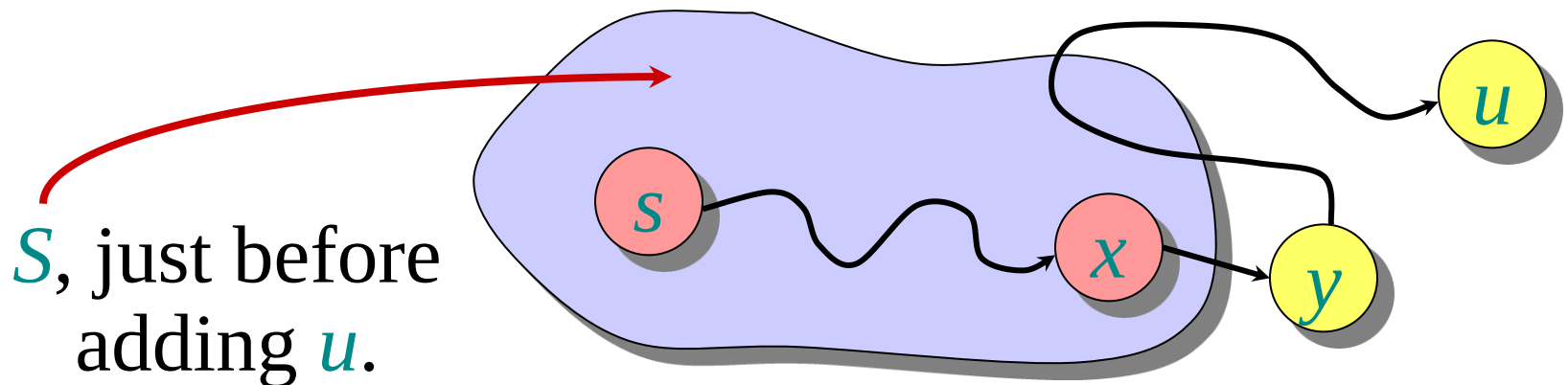
$d[v] < \delta(s, v)$	supposition
$\leq \delta(s, u) + \delta(u, v)$	triangle inequality
$\leq \delta(s, u) + w(u, v)$	sh. path $\leq$ specific path
$\leq d[u] + w(u, v)$	v is first violation

Contradiction. 

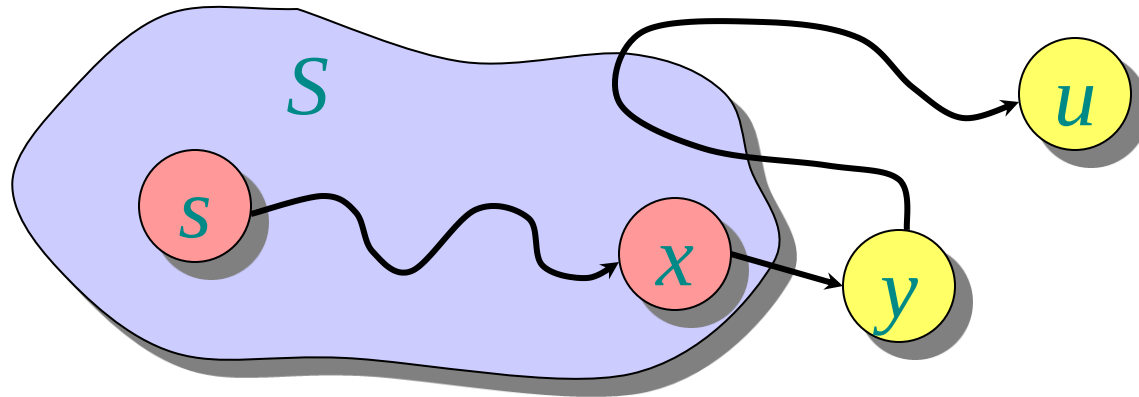
Correctness — Part II

Theorem. Dijkstra's algorithm terminates with $d[v] = \delta(s, v)$ for all $v \in V$.

Proof. It suffices to show that $d[v] = \delta(s, v)$ for every $v \in V$ when v is added to S . Suppose u is the first vertex added to S for which $d[u] \neq \delta(s, u)$. Let y be the first vertex in $V - S$ along a shortest path from s to u , and let x be its predecessor:



Correctness — Part II (continued)



Since u is the first vertex violating the claimed invariant, we have $d[x] = \delta(s, x)$. Since subpaths of shortest paths are shortest paths, it follows that $d[y]$ was set to $\delta(s, x) + w(x, y) = \delta(s, y)$ when (x, y) was relaxed just after x was added to S . Consequently, we have $d[y] = \delta(s, y) \leq \delta(s, u) \leq d[u]$. But, $d[u] \leq d[y]$ by our choice of u , and hence $d[y] = \delta(s, y) = \delta(s, u) = d[u]$. Contradiction. $\square$

Analysis of Dijkstra

$|V|$ times { while $Q \neq \emptyset$
do $u \leftarrow \text{EXTRACT-MIN}(Q)$
 $S \leftarrow S \cup \{u\}$
degree(u) times { for each $v \in \text{Adj}[u]$
do if $d[v] > d[u] + w(u, v)$
then $d[v] \leftarrow d[u] + w(u, v)$

Handshaking Lemma $\Rightarrow \Theta(E)$ implicit DECREASE-KEY's.

$$\text{Time} = \Theta(V) \cdot T_{\text{EXTRACT-MIN}} + \Theta(E) \cdot T_{\text{DECREASE-KEY}}$$

Note: Same formula as in the analysis of Prim's minimum spanning tree algorithm.

Analysis of Dijkstra (continued)

$$\text{Time} = \Theta(V) \cdot T_{\text{EXTRACT-MIN}} + \Theta(E) \cdot T_{\text{DECREASE-KEY}}$$

Q	$T_{\text{EXTRACT-MIN}}$	$T_{\text{DECREASE-KEY}}$	Total
array	$O(V)$	$O(1)$	$O(V^2)$
binary heap	$O(\lg V)$	$O(\lg V)$	$O(E \lg V)$
Fibonacci heap	$O(\lg V)$ amortized	$O(1)$ amortized	$O(E + V \lg V)$ worst case

Unweighted graphs

Suppose $w(u, v) = 1$ for all $(u, v) \in E$. Can the code for Dijkstra be improved?

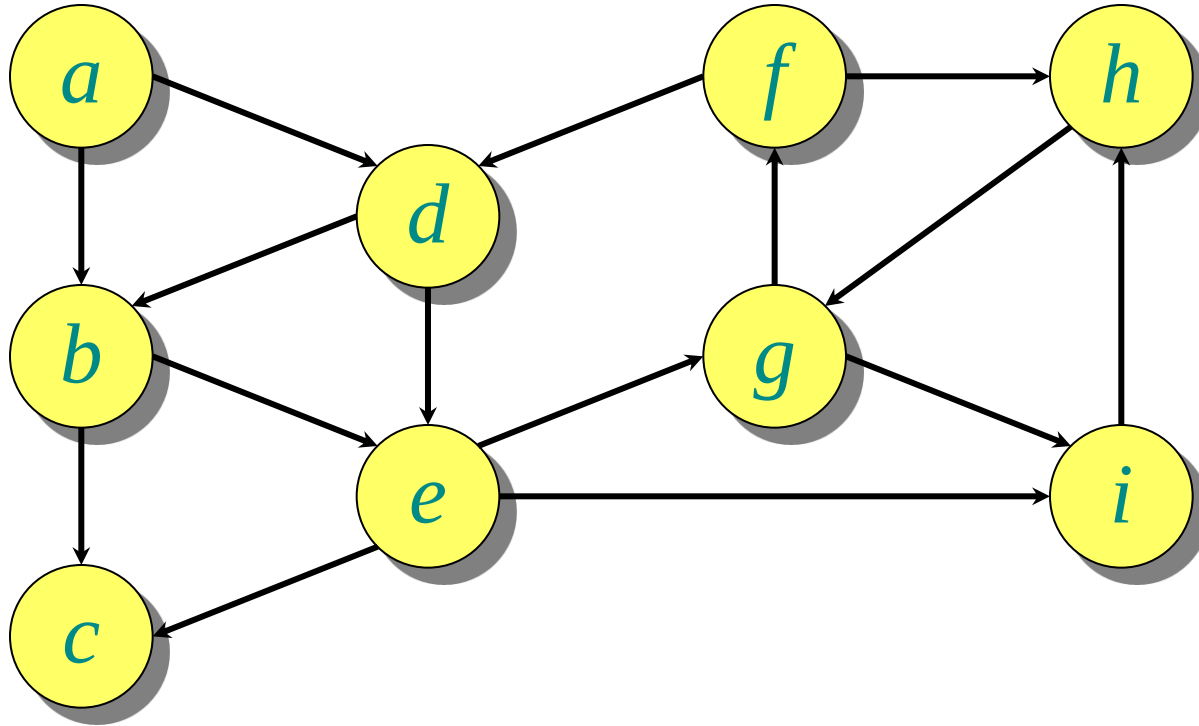
- Use a simple FIFO queue instead of a priority queue.

- ***Breadth-first search***

```
while  $Q \neq \emptyset$ 
do  $u \leftarrow \text{DEQUEUE}(Q)$ 
  for each  $v \in \text{Adj}[u]$ 
    do if  $d[v] = \infty$ 
      then  $d[v] \leftarrow d[u] + 1$ 
          ENQUEUE( $Q, v$ )
```

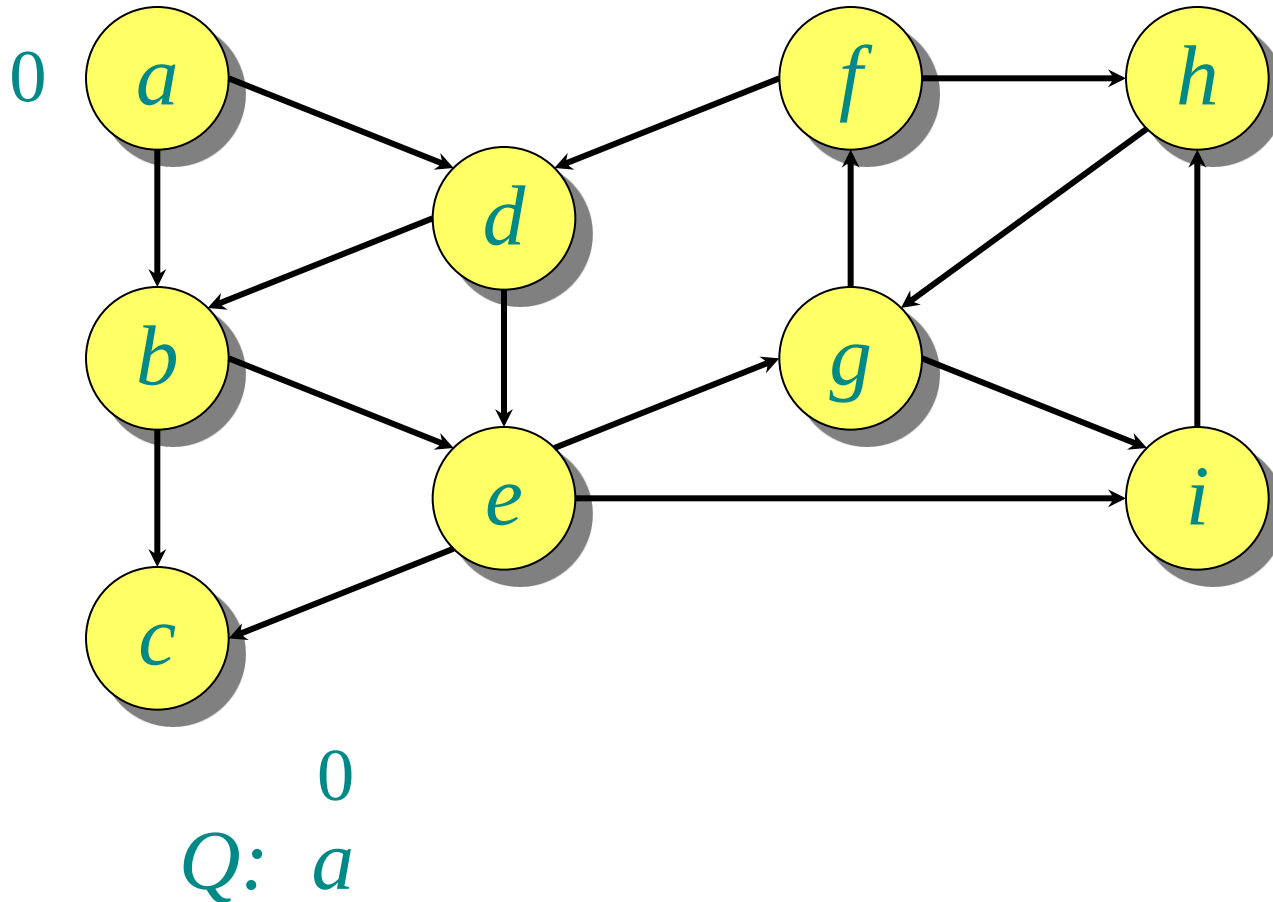
Analysis: Time = $O(V + E)$.

Example of breadth-first search

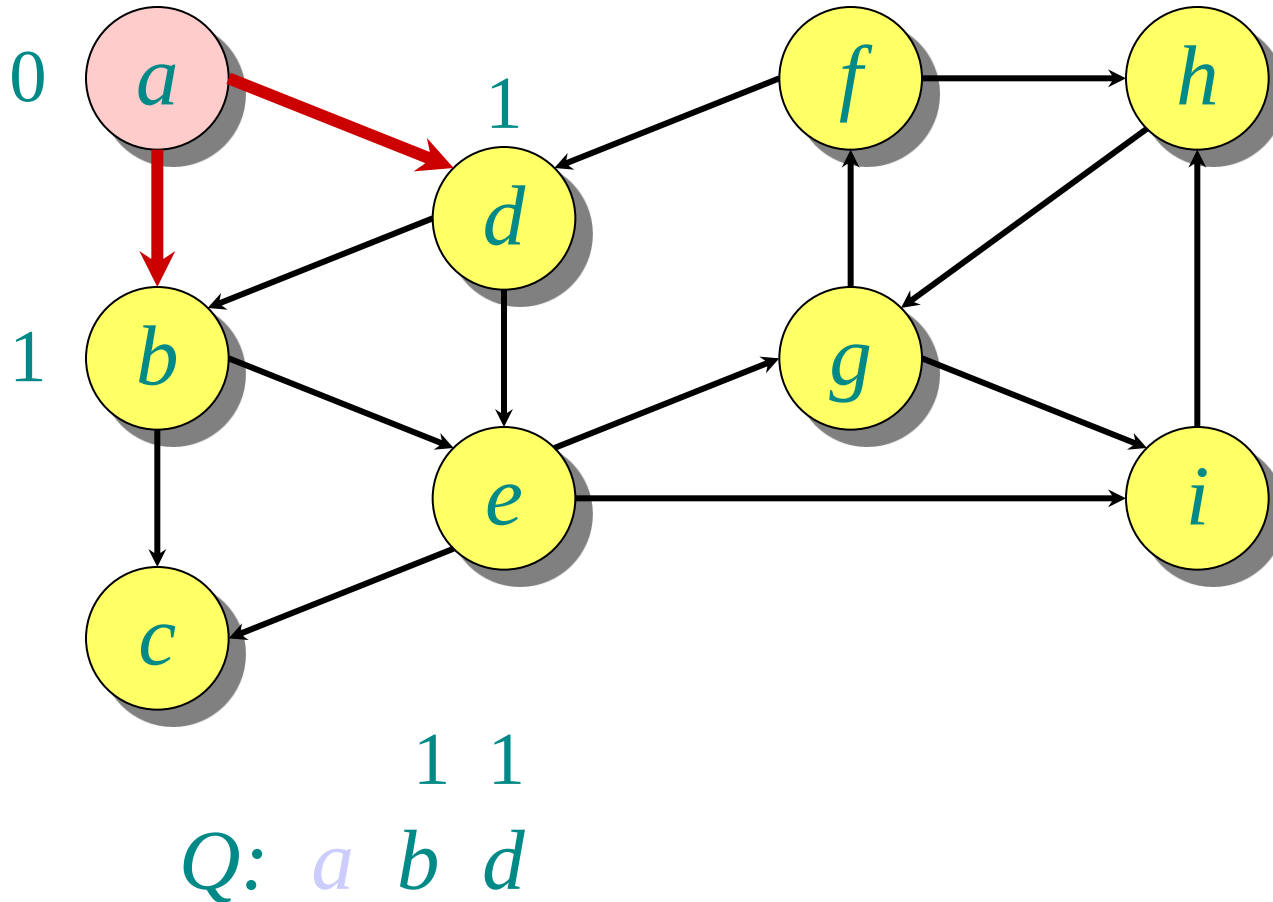


Q:

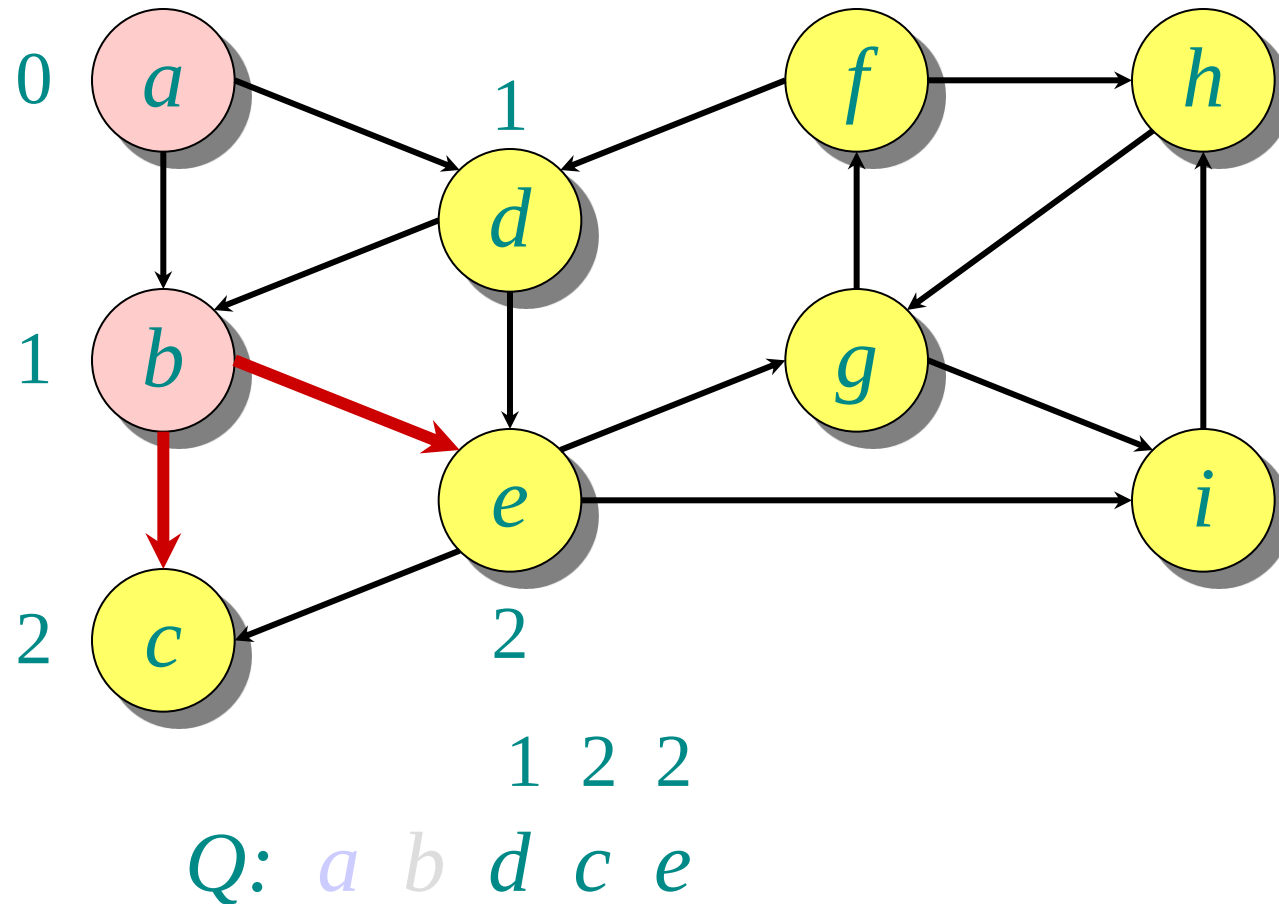
Example of breadth-first search



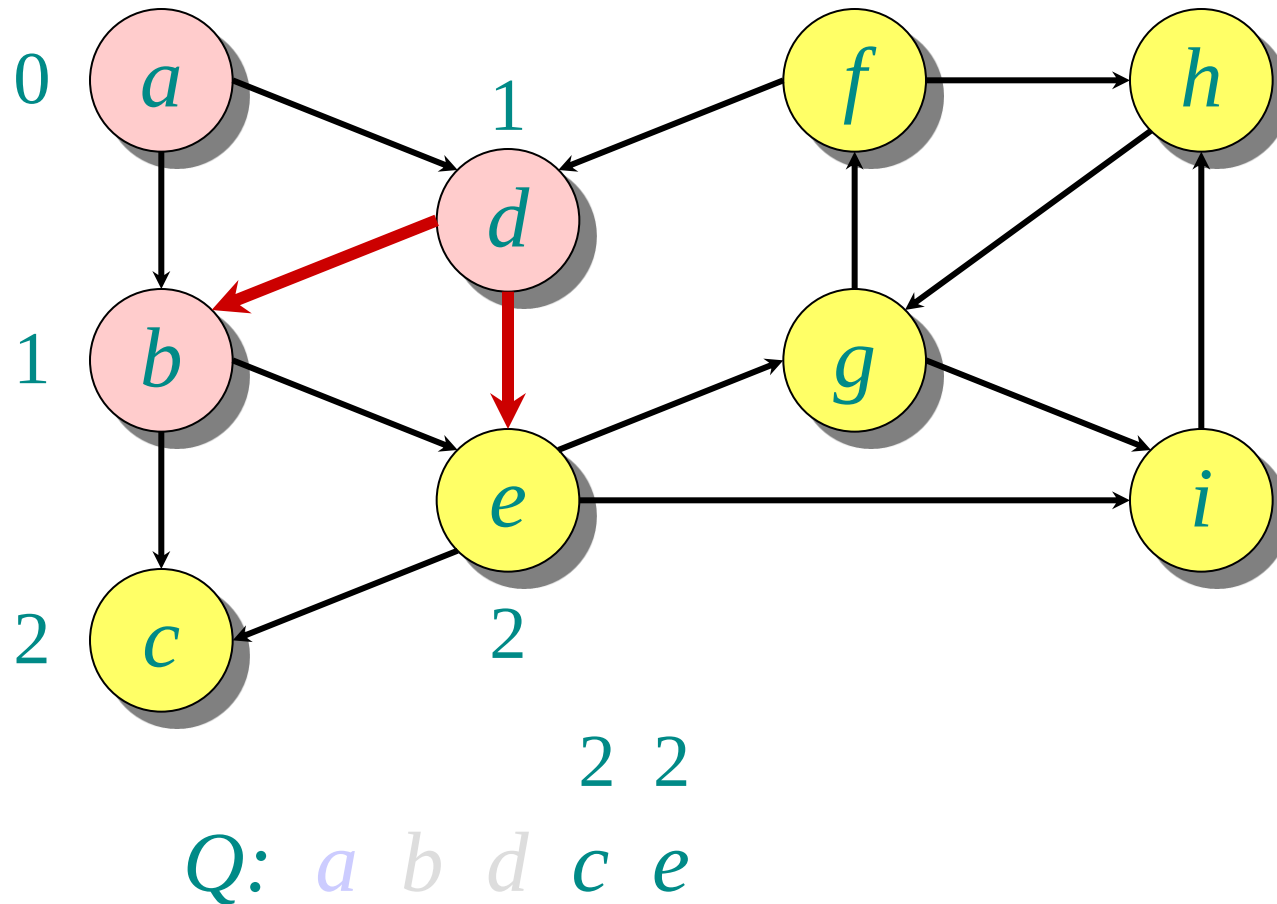
Example of breadth-first search



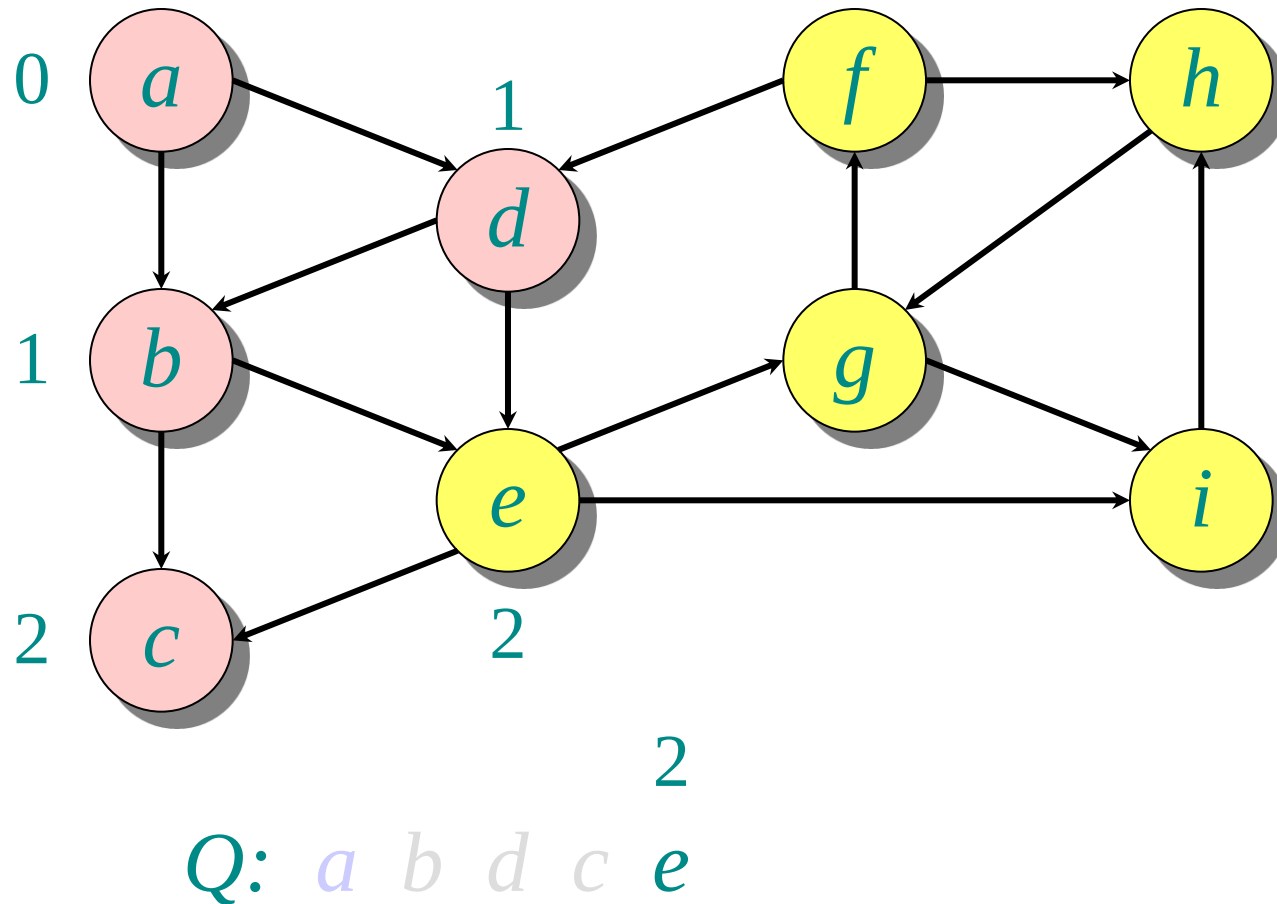
Example of breadth-first search



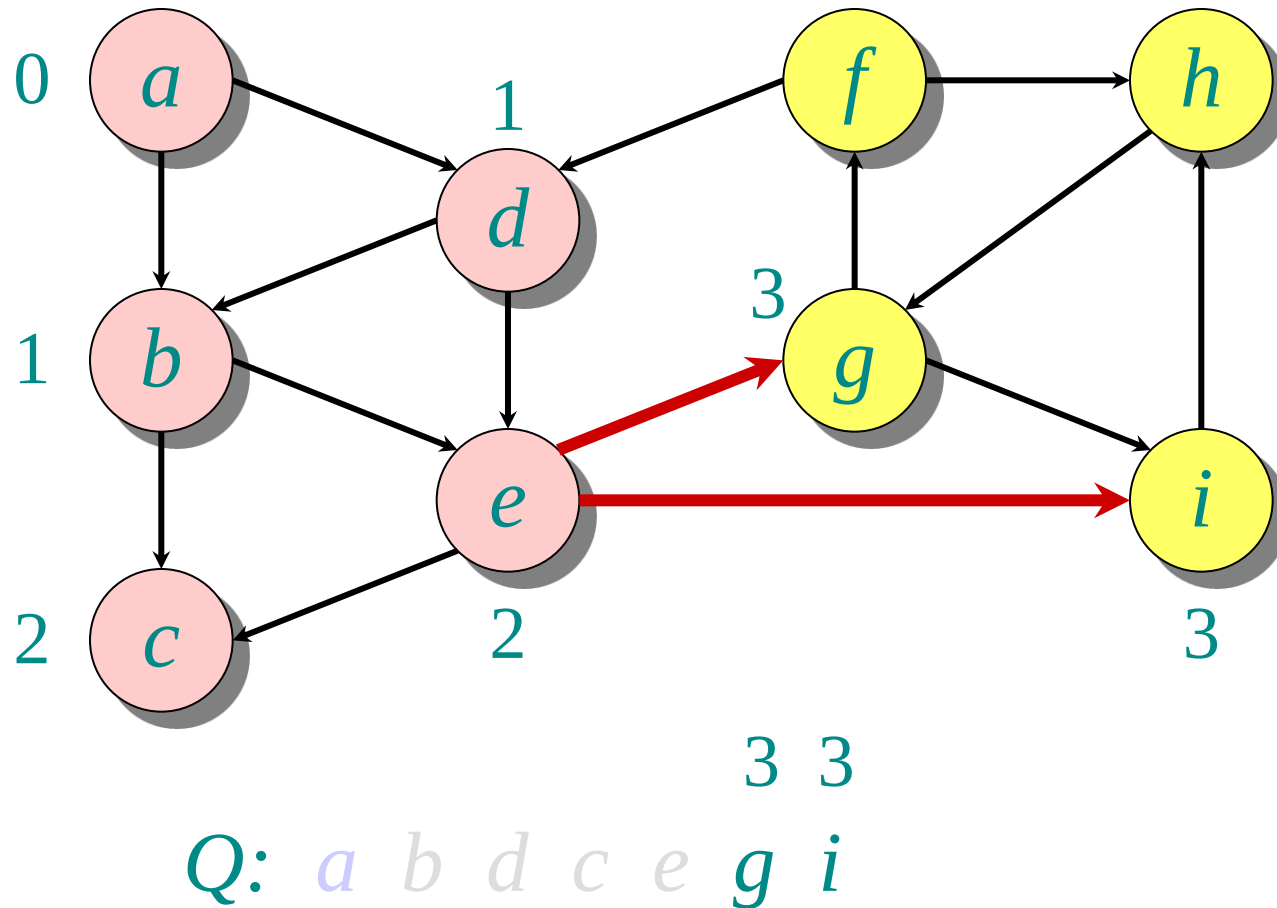
Example of breadth-first search



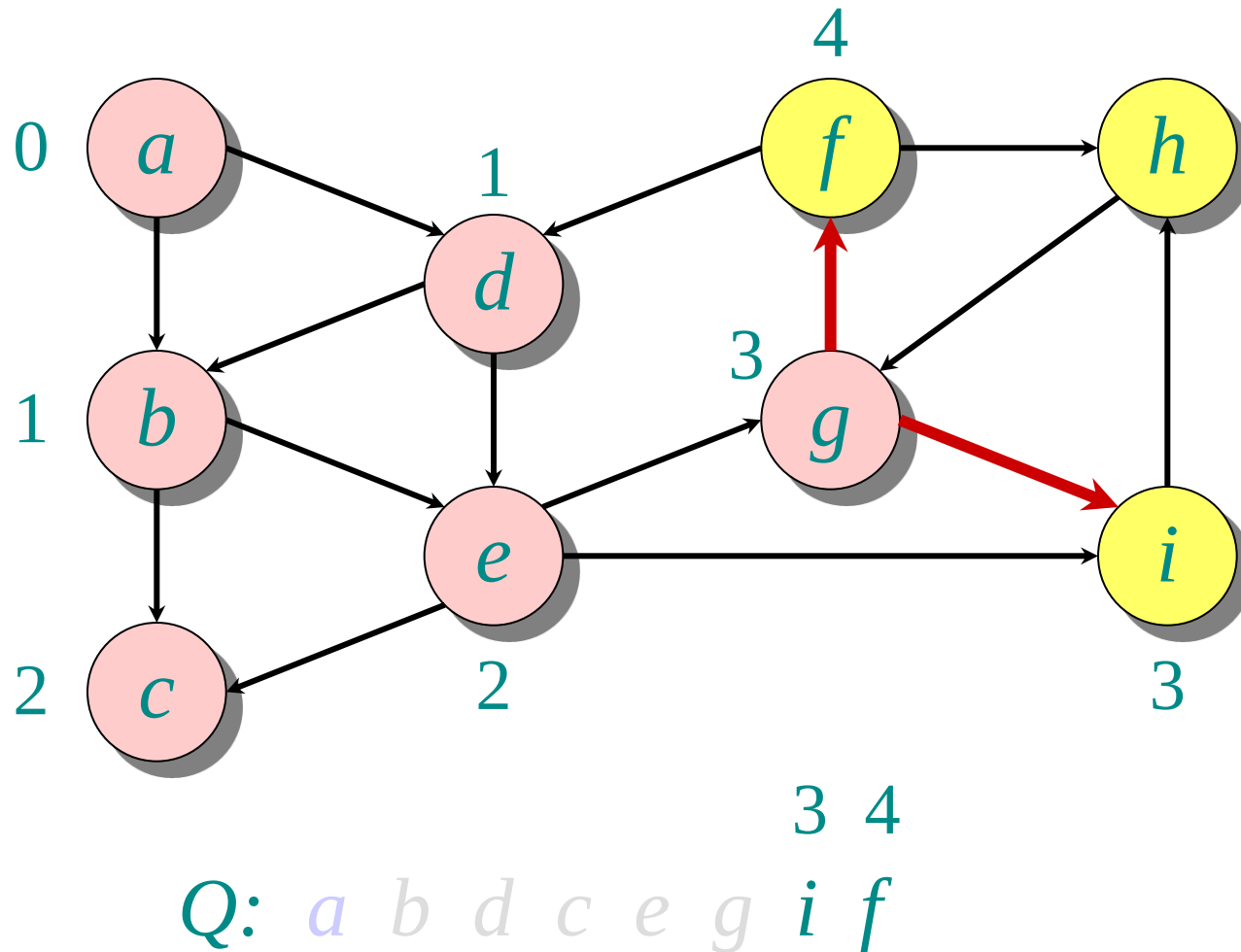
Example of breadth-first search



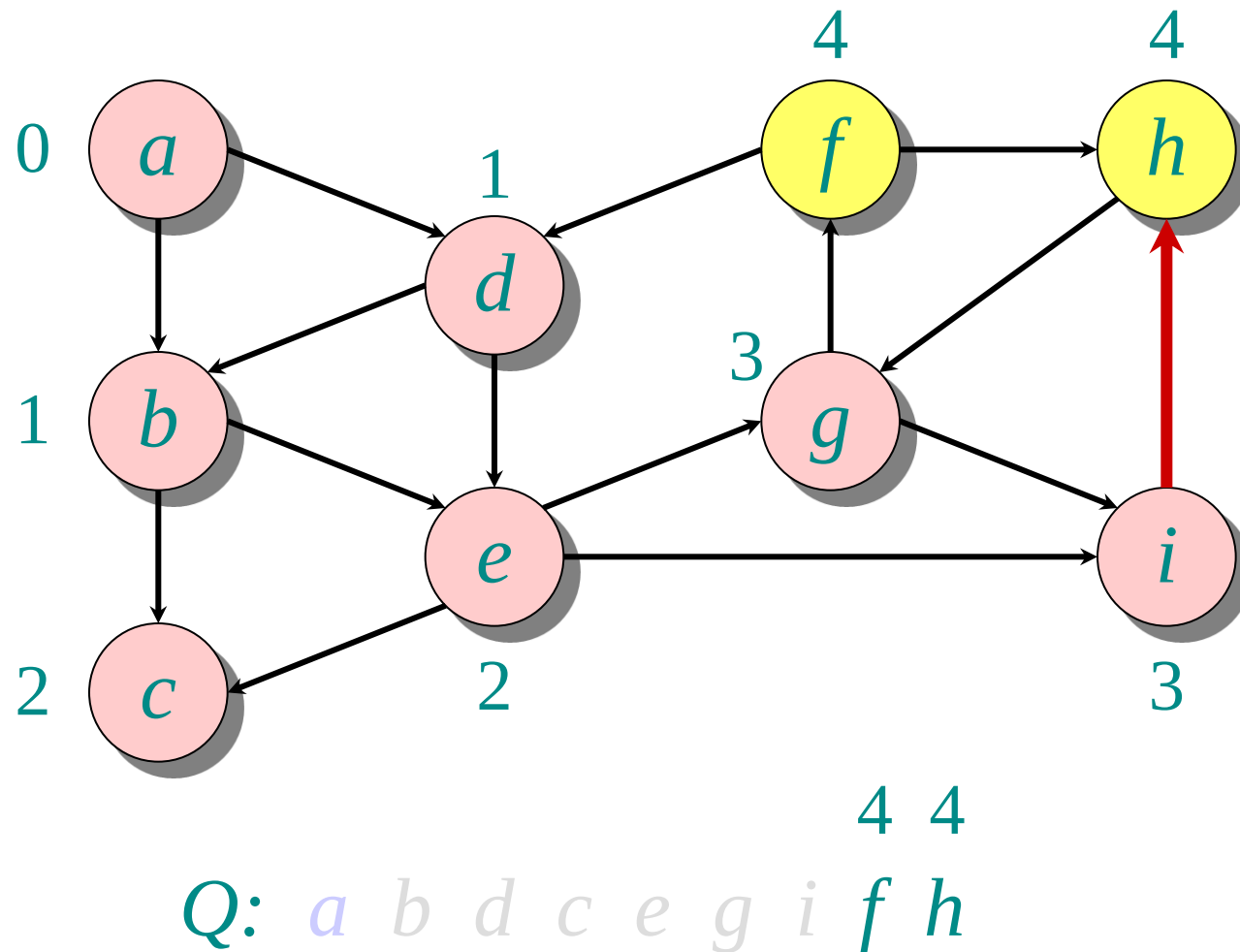
Example of breadth-first search



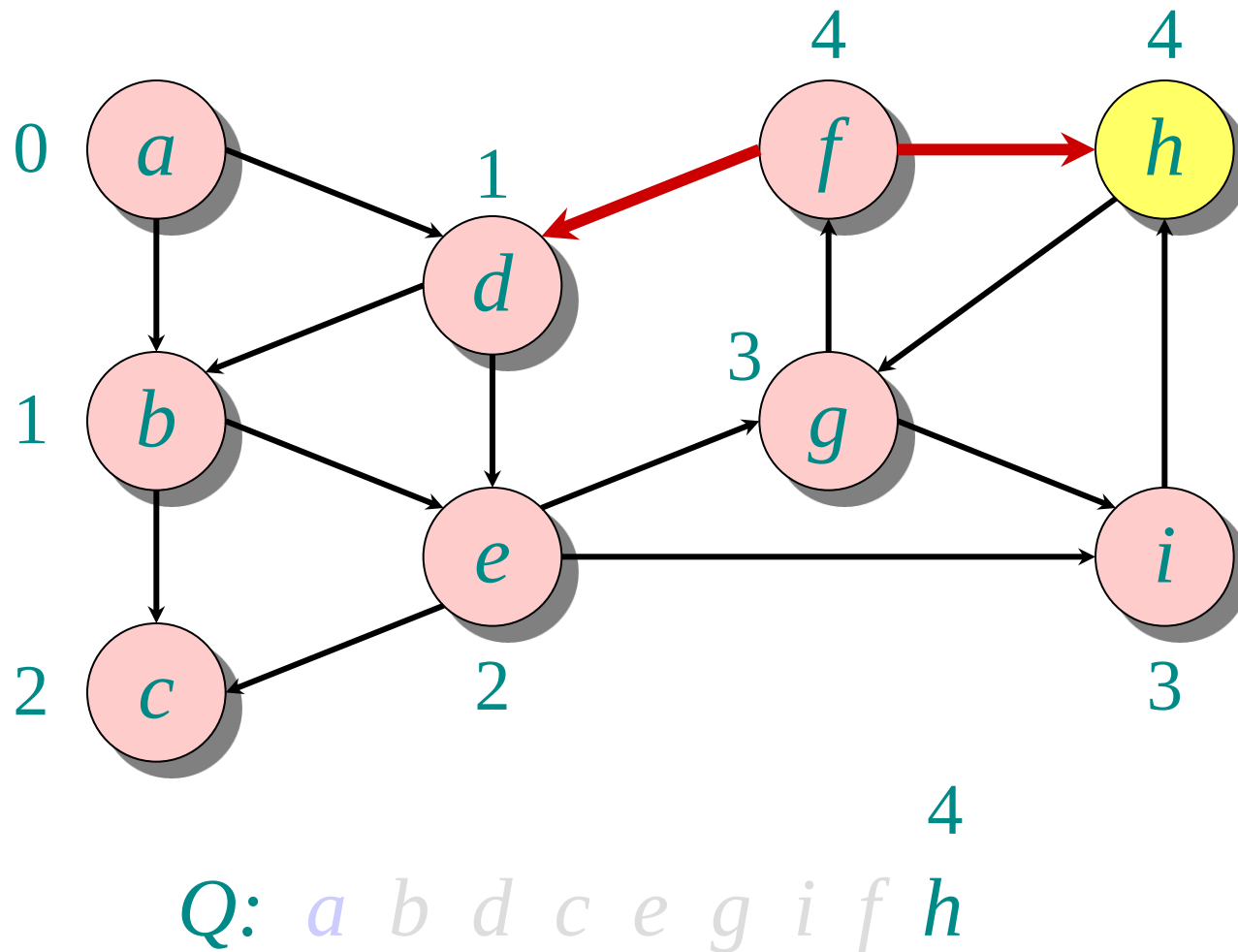
Example of breadth-first search



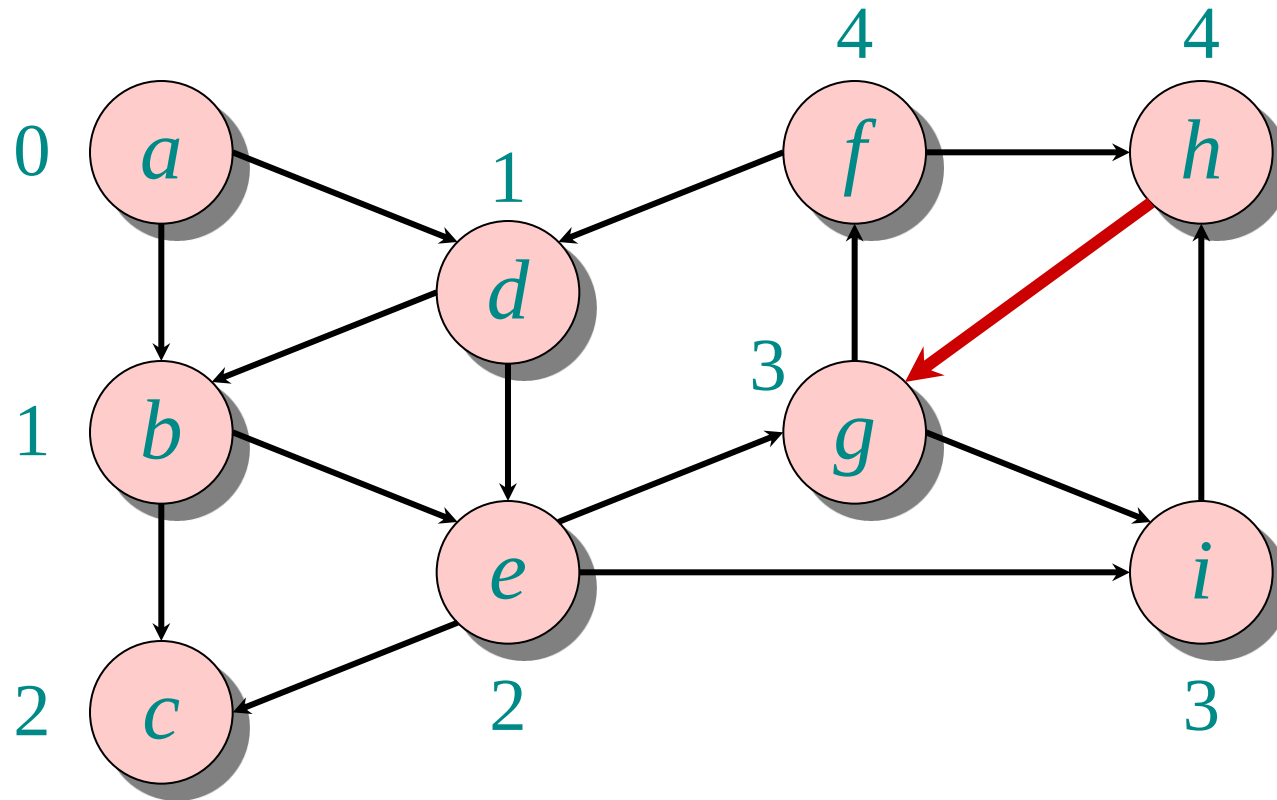
Example of breadth-first search



Example of breadth-first search

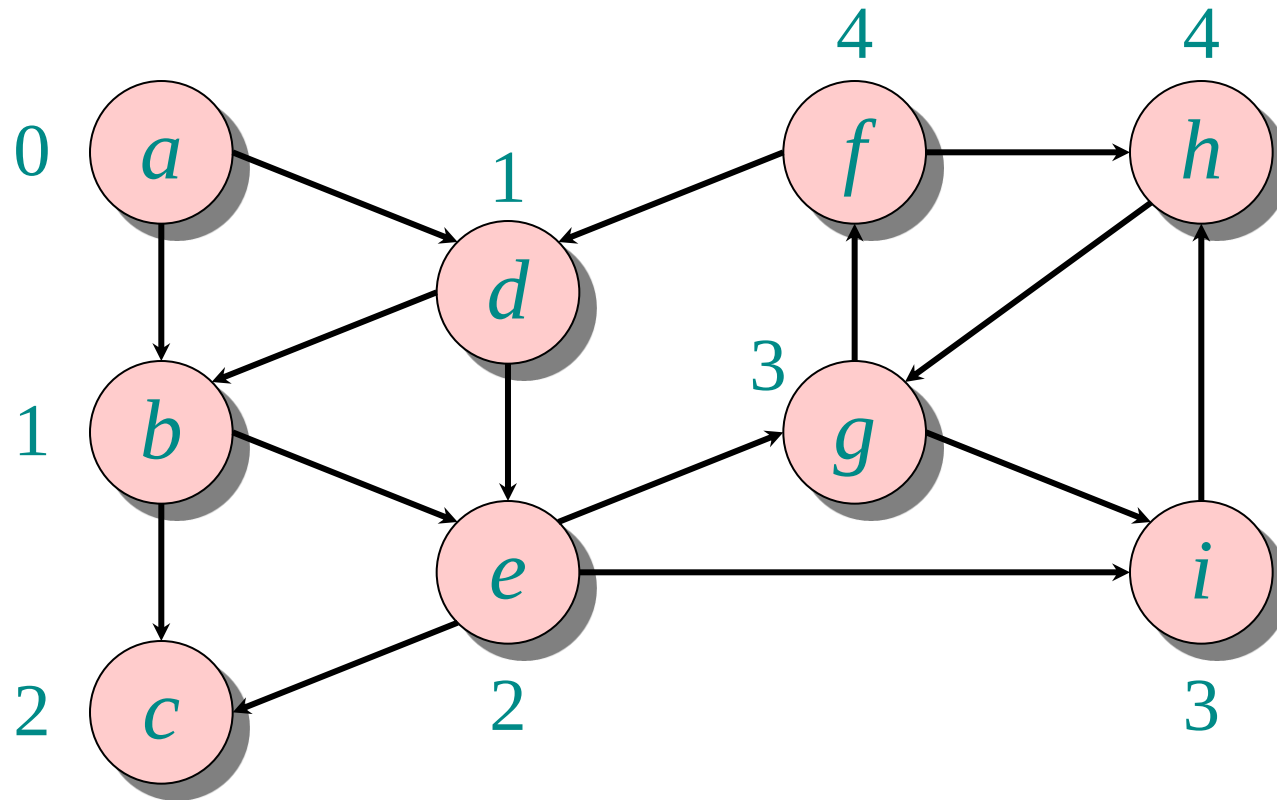


Example of breadth-first search



Q: *a* *b* *d* *c* *e* *g* *i* *f* *h*

Example of breadth-first search



Correctness of BFS

```
while  $Q \neq \emptyset$ 
  do  $u \leftarrow \text{DEQUEUE}(Q)$ 
    for each  $v \in \text{Adj}[u]$ 
      do if  $d[v] = \infty$ 
        then  $d[v] \leftarrow d[u] + 1$ 
           ENQUEUE( $Q, v$ )
```

Key idea:

The FIFO Q in breadth-first search mimics the priority queue Q in Dijkstra.

- **Invariant:** v comes after u in Q implies that $d[v] = d[u]$ or $d[v] = d[u] + 1$.